Cloudera Runtime 7.1.6

CDS 3.1 for GPUs

Date published: 2021-02-29
Date modified: 2021-03-19

https://docs.cloudera.com/

Legal Notice

Contents

CDS 3.1 for GPUs

CDS 3.1 for GPUs is an add-on service that enables you to take advantage of the RAPIDS Accelerator for Apache Spark to accelerate Apache Spark 3 performance on existing CDP Private Cloud Base clusters.

On CDP Private Cloud Base, a Spark 3 service (including CDS 3.1 for GPUs) can coexist with the existing Spark 2 service. The configurations of the two services do not conflict and both services use the same YARN service. The port of the Spark History Server is 18088 for Spark 2 and 18089 for Spark 3.

Unsupported Features:

This release does not support the following features:

- Phoenix Connector
- SparkR
- Hive Warehouse Connector
- Kudu
- Oozie
- Zeppelin

CDS 3.1 for GPUs requirements

CDS 3.1 for GPUs has the following requirements:

Hardware requirements

CDS 3.1 for GPUs requires cluster hosts with NVIDIA Pascal™ or better GPUs, with a compute capability rating of 6.0 or higher.

For more information, see Getting Started at the RAPIDS website.

Cloudera and NVIDIA recommend using NVIDIA-certified systems. For more information, see NVIDIA-Certified Systems in the NVIDIA GPU Cloud documentation.

Software requirements

Each cluster host with a GPU must have the following software installed:

Java

JDK 8 or JDK 11. Cloudera recommends using JDK 8, as most testing has been done with JDK 8. Remove other JDK versions from all cluster and gateway hosts to ensure proper operation.

Scala

Scala 2.12

Python

Python 3.6 and higher

GPU drivers and CUDA toolkit

GPU driver v450.80.02 or higher

CUDA version 11.0 or higher

Download and install the CUDA Toolkit for your operating system. The toolkit installer also provides the option to install the GPU driver.

(Optional) UCX

UCX 1.10.1 required

Clusters with Infiniband or RoCE networking require Unified Communication X (UCX) to enable the RAPIDS Shuffle Manager. For instructions on installing UCX native libraries, see (Optional) Installing UCX native libraries.

CDP versions

Important: CDS 3.1 for GPUs is an add-on service for CDP Private Cloud Base, and is only supported with Cloudera Runtime 7.1.6 and higher. It is not supported on CDP Public Cloud.

Supported versions of CDP are described below.

CDS 3.1 for GPUs	Supported CDP Versions
3.1.7270.2000-53	CDP Private Cloud Base with Cloudera Runtime 7.1.6 and higher

The Spark 2 service (included in CDP) can co-exist on the same cluster as a Spark 3 service (including CDS 3.1 for GPUs, installed as a separate parcel). The two services are configured to not conflict, and both run on the same YARN service. CDS 3.1 for GPUs installs and uses its own external shuffle service.

Although Spark 2 and Spark 3 can coexist in the same CDP Private Cloud Base cluster, you cannot use multiple Spark 3 versions simultaneously. All clusters managed by the same Cloudera Manager Server must use exactly the same version of CDS, whether that is CDS Powered by Apache Spark or CDS 3.1 for GPUs.

Installing CDS 3.1 for GPUs

CDS 3.1 for GPUs is distributed as two files: a custom service descriptor file and a parcel, both of which must be installed on the cluster.

Note: Due to the potential for confusion between CDS 3.1 for GPUs and the initialism CSD, references to the custom service descriptor (CSD) file in this documentation use the term service descriptor.

Remove CDS 3 Powered by Apache Spark

CDS 3.1 for GPUs replaces the CDS 3 Powered by Apache Spark parcel and service. If you have already installed CDS 3 Powered by Apache Spark, remove it before continuing. For instructions on removing parcels, see Deactivate and Remove Parcels.

Install CDS 3.1 for GPUs

Note:

Although Spark 2 and Spark 3 can coexist in the same CDP Private Cloud Base cluster, you cannot use multiple Spark 3 versions simultaneously. All clusters managed by the same Cloudera Manager Server must use exactly the same version of CDS, whether that is CDS Powered by Apache Spark or CDS 3.1 for GPUs.

Additionally, CDS 3.1 for GPUs currently does not work if you have configured a custom location for parcel files. The CDS 3.1 for GPUs and Cloudera Runtime parcels must be stored in the default location, /opt/cloudera/parcels.

Follow these steps to install CDS 3.1 for GPUs:

1. Check that all the software prerequisites are satisfied. If not, you might need to upgrade or install other software components first.

2. Install the CDS 3.1 for GPUs service descriptor into Cloudera Manager.
   a. To download the CDS 3.1 for GPUs service descriptor, click the service descriptor link for the version you want to install.
   b. Log on to the Cloudera Manager Server host, and copy the CDS 3.1 for GPUs service descriptor to the location configured for service descriptor files (/opt/cloudera/csd by default).
   c. Set the file ownership of the service descriptor to cloudera-scm:cloudera-scm with permission 644.
   d. Restart the Cloudera Manager Server with the following command:

```
systemctl restart cloudera-scm-server
```

3. In the Cloudera Manager Admin Console, add the CDS 3.1 for GPUs parcel repository to the Remote Parcel Repository URLs in Parcel Settings as described in Parcel Configuration Settings.

   Note: If your Cloudera Manager Server does not have internet access, you can use the CDS 3.1 for GPUs parcel files: put them into a new parcel repository, and then configure the Cloudera Manager Server to target this newly created repository.
4. Download the CDS 3.1 for GPUs parcel, distribute the parcel to the hosts in your cluster, and activate the parcel. For instructions, see Managing Parcels.
5. Add the CDS 3.1 for GPUs service to your cluster, following the procedure documented in Adding a Service.
   a. In the Add a Service wizard, specify any optional dependencies, such as HBase and Hive, or select No Optional Dependencies.
   b. When customizing the role assignments, add a gateway role to every host.
   c. On the Review Changes page, you can enable TLS for the Spark History Server.
   d. Note that the History Server port is 18089 instead of the usual 18088.
   e. Complete the remaining steps in the wizard.
6. Return to the Home page by clicking the Cloudera Manager logo in the upper left corner.
7. Click the stale configuration icon to launch the Stale Configuration wizard and restart the necessary services.

Install the Livy for Spark 3 Service Descriptor

CDS 3.1 for GPUs supports Apache Livy, but it cannot use the included Livy service, which is compatible with only Spark 2. To add and manage a Livy service compatible with Spark 3, you must install a service descriptor for the Livy for Spark 3 service.

1. Install the Livy for Spark 3 service descriptor into Cloudera Manager.
   a. To download the service descriptor, click the Livy for Spark 3 service descriptor link for the version you want to install.
   b. Log on to the Cloudera Manager Server host, and copy the Livy service descriptor to the location configured for service descriptor files (/opt/cloudera/csd by default).
   c. Set the file ownership of the service descriptor to cloudera-scm:cloudera-scm with permissions set to 644.
   d. Restart the Cloudera Manager Server with the following command:

```
systemctl restart cloudera-scm-server
```

2. Add the Livy for Spark 3 service to your cluster, following the procedure documented in Adding a Service.

   Note: The port for Livy for Spark 3 is 28998 instead of the usual 8998.
3. Return to the Home page by clicking the Cloudera Manager logo in the upper left corner.
4. Click the stale configuration icon to launch the Stale Configuration wizard and restart the necessary services.

(Optional) Installing UCX native libraries

The NVIDIA RAPIDS Shuffle Manager is a custom ShuffleManager for Apache Spark that allows fast shuffle block transfers between GPUs in the same host (over PCIe or NVLink) and over the network to remote hosts (over RoCE or Infiniband). The RAPIDS Shuffle Manager is based on Unified Communication X (UCX).

About this task

NVIDIA RAPIDS Shuffle Manager has been shown to accelerate workloads where shuffle is the bottleneck when using the RAPIDS accelerator for Apache Spark. It accomplishes this by using a GPU shuffle cache for fast shuffle writes when shuffle blocks fit in GPU memory, avoiding the cost of writes to host using the built-in Spark Shuffle, a spill framework that will spill to host memory and disk on demand, and UCX as its transport for fast network and peer-to-peer (GPU-to-GPU) transfers.

Cloudera and NVIDIA recommend using the RAPIDS shuffle manager for clusters with Infiniband or RoCE networking.

Procedure

1. Download UCX v1.10.1 for your operating system.
2. Install the downloaded packages using your operating system package manager.

   For CentOS, if you do not have Infiniband or RoCE networking, you only need to install the following packages:

   ```
   ucx-1.10.1-1.el7.x86_64.rpm
   ucx-cuda-1.10.1-1.el7.x86_64.rpm
   ```

   If you have Infiniband or RoCE networking, install the following packages:

   ```
   ucx-1.10.1-1.el7.x86_64.rpm
   ucx-cuda-1.10.1-1.el7.x86_64.rpm
   ucx-rdmacm-1.10.1-1.el7.x86_64.rpm
   ucx-ib-1.10.1-1.el7.x86_64.rpm
   ```

3. Configure Mellanox Infiniband or RoCE networking. For more information, see spark-rapids documentation and Mellanox documentation.

   Note, that:

   - UCX is agnostic to switch model and make.
   - UCX is fully tested on Mellanox RDMA hardware, other hardware may not be compatible. UCX supports TCP when RDMA is not possible otherwise, but it is not recommended as the best performance for UCX is with RDMA hardware.
4. Validate your UCX environment following the instructions provided in the NVIDIA spark-rapids documentation.

5. Before running applications with the RAPIDS Shuffle Manager, make the following configuration changes:
   a) Disable the External Shuffle Service:

   ```
   spark.shuffle.service.enabled=false
   ```

   b) Disable Dynamic Allocation:

   ```
   spark.dynamicAllocation.enabled=false
   ```

   c) Enable the RAPIDS Shuffle Manager:

   ```
   spark.shuffle.manager=com.nvidia.spark.rapids.spark311cdh.RapidsShuffleM
   anager
   ```

   d) Specify the “extraClassPath” Executor:

   ```
   spark.executor.extraClassPath=/opt/cloudera/parcels/SPARK3_RAPIDS/lib/sp
   ark3/rapids-plugin/*
   ```

   e) At a minimum, make the following UCX settings:

   ```
   spark.executorEnv.UCX_ERROR_SIGNALS=
   spark.executorEnv.UCX_MEMTYPE_CACHE=n
   ```

   f) Recommended additional UCX settings:

   ```
   spark.executorEnv.UCX_TLS=cuda_copy,cuda_ipc,rc,tcp
   spark.executorEnv.UCX_RNDV_SCHEME=put_zcopy
   spark.executorEnv.UCX_MAX_RNDV_RAILS=1
   spark.executorEnv.UCX_IB_RX_QUEUE_LEN=1024
   ```

   For more information on environment variables, see the NVIDIA spark-rapids documentation.
6.

Setting up a Yarn role group

Create a Yarn role group so that you can selectively enable GPU usage for nodes with GPUs within your cluster.

About this task

Enabling GPU on YARN through the Enable GPU Usage tickbox operates on cluster-level. Role groups in Yarn enable you to apply settings selectively, to a subset of nodes within your cluster.

Role groups are configured on the service level.

Before you begin

Configure GPU scheduling and isolation

Procedure

1. In Cloudera Manager navigate to Yarn Instances .
2. Create a role group where you can add nodes with GPUs.
   For more information, see Creating a Role Group.
3. Move role instances with GPUs to the group you created.
   On the Configuration tab select the source role group with the hosts you want to move, then click Move Selected Instances To Group and select the role group you created.
   You may need to restart the cluster.

4. Enable GPU usage for the role group.
   a) On the Configuration tab select Categories GPU Management .
   b) Under GPU Usage click Edit Individual Values and select the role group you created.
   c) Click Save Changes.

Running applications with CDS 3.1 for GPUs

With CDS 3.1 for GPUs, you can run Apache Spark 3 applications locally or distributed across a cluster, either by using an interactive shell or by submitting an application. Running Spark applications interactively is commonly performed during the data-exploration phase and for ad hoc analysis.

The Spark 3 job commands

With Spark 3, you use slightly different command names than with Spark 2, so that you can run both versions of Spark side-by-side without conflicts:

- spark3-submit instead of spark-submit.
- spark3-shell instead of spark-shell.
- pyspark3 instead of pyspark.

For development and test purposes, you can also configure an alias on each host so that invoking the Spark 2 command name runs the corresponding Spark 3 executable.

Running a Spark job with CDS 3.1 for GPUs

1. Log on to the node where you want to run the job.
2. Run the following command to launch spark3-shell:

```
spark3-shell  --conf "spark.rapids.sql.enabled=true" \
              --conf "spark.executor.memoryOverhead=5g"
```

where
--conf spark.rapids.sql.enabled=true

enables the following environment variables for GPUs:

```
"spark.task.resource.gpu.amount" - sets GPU resource amount per
task

"spark.rapids.sql.concurrentGpuTasks" - sets the number of con
current tasks per GPU
"spark.sql.files.maxPartitionBytes" - sets the input partition
 size for DataSource API, The recommended value is "256m".
"spark.locality.wait" - controls how long Spark waits to obtain b
etter locality for tasks.
"spark.sql.adaptive.enabled" - enables Adaptive Query Execution.
"spark.rapids.memory.pinnedPool.size" - sets the amount of pinned
 memory allocated per host.
"spark.sql.adaptive.advisoryPartitionSizeInBytes" - sets the ad
visory size in bytes of the shuffle partition during adaptive op
timization.
```

For example,

```
$SPARK_HOME/bin/spark3-shell \
```

```
        --conf spark.task.resource.gpu.amount=2 \
        --conf spark.rapids.sql.concurrentGpuTasks=2 \
        --conf spark.sql.files.maxPartitionBytes=256m \
        --conf spark.locality.wait=0s \
        --conf spark.sql.adaptive.enabled=true \
        --conf spark.rapids.memory.pinnedPool.size=2G \
        --conf spark.sql.adaptive.advisoryPartitionSizeInBytes=1g
```

--conf "spark.executor.memoryOverhead=5g"

sets the amount of additional memory to be allocated per executor process

Note: cuDF uses a Just-In-Time (JIT) compilation approach for some kernels, and the JIT process can add a few seconds to query wall-clock time. You are recommended to set a JIT cache path to speed up subsequent invocations with: --conf spark.executorEnv.LIBCUDF_KERNEL_CACHE_PATH=[local pa th]. The path should be local to the executor (not HDFS) and not shared between different cluster users in a multi-tenant environment. For example, the path may be in /tmp: (/tmp/cudf-[***USER***]). If the/tmp directory is not writable, consult your system administrator to find a path that is.

You can override these configuration settings both from the command line and from code. For more information on environment variables, see the NVIDIA spark-rapids documentation and the Spark SQL Performance Tuning Guide.

3. Run a job in spark3-shell.

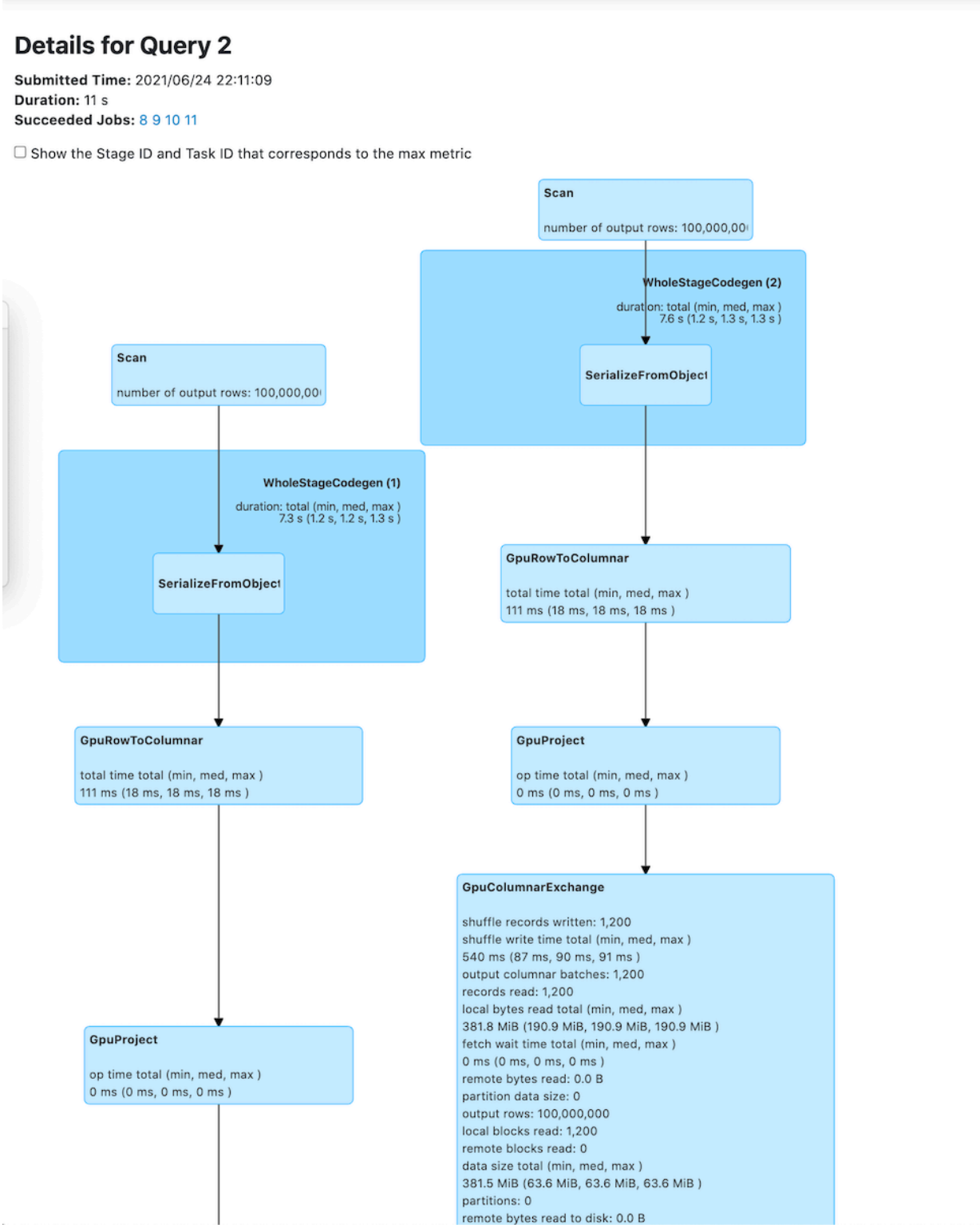
   For example:

```
scala> val df = sc.makeRDD(1 to 100000000, 6).toDF
df: org.apache.spark.sql.DataFrame = [value: int]

scala>val df2 = sc.makeRDD(1 to 100000000, 6).toDF
df2: org.apache.spark.sql.DataFrame = [value: int]
scala> df.select($"value" as "a").join(df2select($"value" as "b"), $"a"
 === $"b").count
res0: Long = 100000000
```

4. You can verify that the job run used GPUs, by logging on to the Yarn UI v2 to review the execution plan and the performance of your spark3-shell application:

   Select the Applications tab then select your [spark3-shell application]. Select ApplicationMaster SQL count at <console>:28 to see the execution plan.



Accessing the Spark 3 History Server

The Spark 3 history server is available on port 18089, rather than port 18088 as with the Spark 2 history server.

CDS 3.1 for GPUs version and download information

The following sections provide links to the parcel and service descriptor files for CDS 3.1 for GPUs.

CDS 3.1 for GPUs versions available for download

Table 1: Available CDS for GPUs versions

Version	Custom Service Descriptors	Parcel Repository
3.1.7270.2000-53	• SPARK3_ON_YARN-3.1.1.3.1.7270.2000-53.jar • LIVY_FOR_SPARK3-0.6.0.3.1.7270.2000-53.jar	https://archive.cloudera.com/p/spark3-rapids/3.x/parcels/